

# Creating your first Microsoft Sentinel Notebook

techcommunity.microsoft.com/t5/microsoft-sentinel-blog/creating-your-first-microsoft-sentinel-notebook/ba-p/2977745

November 18, 2021

```
[59] 1 %pip install requests
      ✓ 3 sec
... Requirement already satisfied: requests in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (2.26.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (from requests) (1.25.11)
Requirement already satisfied: charset-normalizer~=2.0.0 in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (from requests) (2.0.7)
Requirement already satisfied: idna<4,>=2.5 in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (from requests) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (from requests) (2021.5.30)
WARNING: You are using pip version 21.3; however, version 21.3.1 is available.
You should consider upgrading via the '/anaconda/envs/azureml_py36/bin/python -m pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.
```

*This installment is part of a broader learning series to help you become a Jupyter Notebook ninja in Microsoft Sentinel. The installments will be bite-sized to enable you to easily digest the new content.*

- **Part 1:** [What are notebooks and when do you need them?](#)
- **Part 2:** [How to get started with notebooks and tour the features](#)
- **Part 3:** [Overview of the pre-built notebooks and how to use them](#)
- **Part 3.5:** [Using Code Snippets to build your own Sentinel Notebooks](#)
- **Part 4:** How to create your own notebooks from scratch and how to customize the existing ones - **this post**

**KNOWLEDGE CHECK:** And, once you've completed all of the parts of this series, you can take the [Knowledge Check](#). If you score 80% or more in the [Knowledge Check](#), you can expect your very own Notebooks Ninja participation certificate from us.

Jupyter Notebooks are a fantastic resource for security analysts, providing a range of powerful and flexible capabilities. Microsoft Sentinel's integration with Notebooks can provide a quick and straightforward way for security analysts to use Notebooks, however for those new to Notebooks and coding they can be a little daunting.

In this blog we will cover some of the basics of creating your first Microsoft Sentinel Notebook using Python, including how to troubleshoot some common issues you may come across.

- Installing and importing packages in Python
- Installing and importing MSTICPy
- Setting up MSTICPy's config file
- Getting data from Microsoft Sentinel
- Working with data

- Enriching results with external data sources
- Visualizations with MSTICPy

Before we begin, make sure to familiarize yourself with Notebooks in Microsoft Sentinel via Azure Machine Learning.

### Use Jupyter Notebooks to hunt for security threats

If you wish to learn more about this topic, we are running introductory training on December 16th, 2021: Become a Jupyter Notebooks Ninja – MSTICPy Fundamentals to Build Your Own Notebooks. [Sign Up Here](#)

## Installing and Importing Packages in Python

One of the important things about using Python in Notebooks is that you can install and use code libraries (referred to as packages) created by others, allowing you to access the functionality they provide without having to code them yourself.

There are several ways to install Python packages depending on how you want to find and access the packages, however the simplest and easiest is using pip.

Pip (<https://pypi.org/project/pip/>) is the package installer for Python and makes finding and installing Python packages simple.

You can use pip to install packages via the command line, or if you are using a Notebook, directly in a Notebook cell. Installing directly in a Notebook is often preferred as it ensures that you are installing the package in the same Python environment the Notebook is being executed in. To install via a Notebook code cell, we need to use `%pip` followed by install and the package name. e.g.:

```
%pip install requests
```

```

1 %pip install requests
[59] ✓ 3 sec

... Requirement already satisfied: requests in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (2.26.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (from requests) (1.25.11)
Requirement already satisfied: charset-normalizer~=2.0.0 in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (from requests) (2.0.7)
Requirement already satisfied: idna<4,>=2.5 in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (from requests) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /anaconda/envs/azureml_py36/lib/python3.6/site-packages (from requests) (2021.5.30)
WARNING: You are using pip version 21.3; however, version 21.3.1 is available.
You should consider upgrading via the '/anaconda/envs/azureml_py36/bin/python -m pip install --upgrade pip' command.
Note: you may need to restart the kernel to use updated packages.

```

Notebook output of running `%pip install requests`

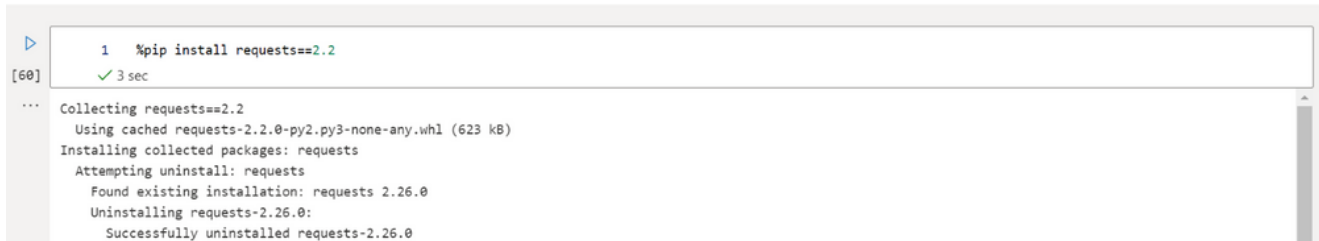
**Note:** `%pip` is what is called a magic function in Jupyter. This tells the Notebook to use pip to install the package in the Notebooks compute environment.

If you already have a package installed but you want to update to the latest version, you can add the `--upgrade` parameter to the command used:

```
%pip install -upgrade requests
```

You may also want to install a specific version of a package. This can be done by specifying the version number.

```
%pip install requests==2.22.0
```



```
1 %pip install requests==2.2
[60] ✓ 3 sec
...
Collecting requests==2.2
Using cached requests-2.2.0-py2.py3-none-any.whl (623 kB)
Installing collected packages: requests
Attempting uninstall: requests
Found existing installation: requests 2.26.0
Uninstalling requests-2.26.0:
Successfully uninstalled requests-2.26.0
```

Output of running `%pip install requests==2.22.0`

**Note:** Once you have installed a package it is recommended to restart the Notebook kernel, this will ensure that when you import the package you will be using the latest version. This is not necessary with newly installed packages but is important when

**Note:** During installation of packages you may see some warnings related to package dependencies. This is because some packages have requirements on other packages being installed and sometimes these requirements can have conflicts (i.e., package 1 requires package A version 1.1 but package 2 also requires package A but version 1.2). We try to avoid conflicts as much as possible with our Notebooks but sometimes these can occur. You can usually run the Notebook without the conflicts affecting you. However, if you encounter a problem with a pre-made Microsoft Sentinel Notebook, please report this at via [GitHub](#).

Once a package is installed, you need to import the package before it can be used. This is done with the ``import`` statement.

There are 2 ways to import things in Python:

- ``import <package>`` - this will do a standard import of the package
- ``from <package> import <item>`` - this imports a specific item from the package

You can also import packages and rename them for ease when calling them later:

```
`import <package> as <alias>`
```

```
import pandas as pd
```

**Troubleshooting Tip:** Some packages do not use the same name for installation and import. You may need to check package documentation to ensure you are importing correctly.

For example, the popular Machine Learning tool package scikit-learn is installed with:

```
%pip install scikit-learn
```

However, it is imported with:

```
import sklearn
```

## Installing and Importing MSTICPy

---

Now that we know how to install and import packages, we can install packages that will be useful to us in creating our Notebook. MSTICPy is a package created by the Microsoft Threat Intelligence Center (MSTIC) and provides a range of tools to make security analysis and investigations in Notebooks quicker and easier. You can find out more about MSTICPy here:

[ReadTheDocs - MSTICPy](#)

We can now install MSTICPy. To make sure we get the latest version if we already have it installed, we are going to use the `--upgrade` parameter.

```
%pip install --upgrade msticpy
```

Now we could import MSTICPy with `import msticpy` however it is a big package with a lot of features, so to make it easier we have a function called `init_notebook` that conducts several checks to make sure the environment is good, handles key imports and set up for us.

```
import msticpy
msticpy.init_notebook(globals())
```

```
1 import msticpy
2
3 msticpy.init_notebook(globals())
[103] ✓ 2 sec
... True
```

---

**Starting Notebook initialization...**

**Starting notebook pre-checks...**

Checking Python kernel version...Recommended: switch to using the 'Python 3.8 - AzureML' notebook kernel if this is available.Info: Python kernel version 3.6.9 - OK  
Checking msticpy version...  
Info: msticpy version 1.4.5 (>= 1.4.5) - OK

**Notebook pre-checks complete.**

Processing imports...  
Imported: pd (pandas), IPython.get\_ipython, IPython.display.display, IPython.display.HTML, IPython.display.Markdown, widgets (ipywidgets), pathlib.Path, plt (matplotlib.pyplot), matplotlib.MatplotlibDeprecationWarning, sns (seaborn), np (numpy), msticpy, msticpy.data.QueryProvider, msticpy.nbtools.foliummap.FoliumMap, msticpy.common.utility.md, msticpy.common.utility.md\_warn, msticpy.common.wsconfig.WorkspaceConfig, msticpy.datamodel.pivot.Pivot, msticpy.datamodel.entities, msticpy.vis.mp\_pandas\_plot

Checking configuration...  
Azure CLI was detected but the token has expired. For Azure CLI single sign-on, please sign in using '!az login'.  
For more information see [Caching credentials with Azure CLI](#)  
[Setting notebook options...](#)

[Notebook initialization complete](#)

Notebook output of running previous code cell.

## Setting up MSTICPy's Config File

MSTICPy can handle connections to a variety of data sources and services, including Microsoft Sentinel. As such it needs to handle several configuration details and credentials, things such as the Microsoft Sentinel workspaces you want to get data from, or API (Application Programming Interfaces) keys for external services such as Virus Total.

To make it easier to manage and re-use the configuration and credentials for these things MSTICPy has its own config file that holds these items - `msticpyconfig.yaml`.

The first time you use MSTICPy you need to populate your msticpyconfig.yaml file. This is a one-time activity once you have created it, you can simply re-use in future. To help with the set-up we have created several Notebook widgets to help you populate the file.

Azure Sentinel workspace settings

Workspaces	Name	WorkspaceId	TenantId	SubscriptionId	ResourceGroup
CS	CyberSecDemo	8ecf8077-cf51-4820-aadd-14040956f35d	72f988bf-86f1-41af-91ab-2d7cd011db47		
Cent					
Default					
Web1					
Web2					

Buttons: Add, Delete, Set as default, Update

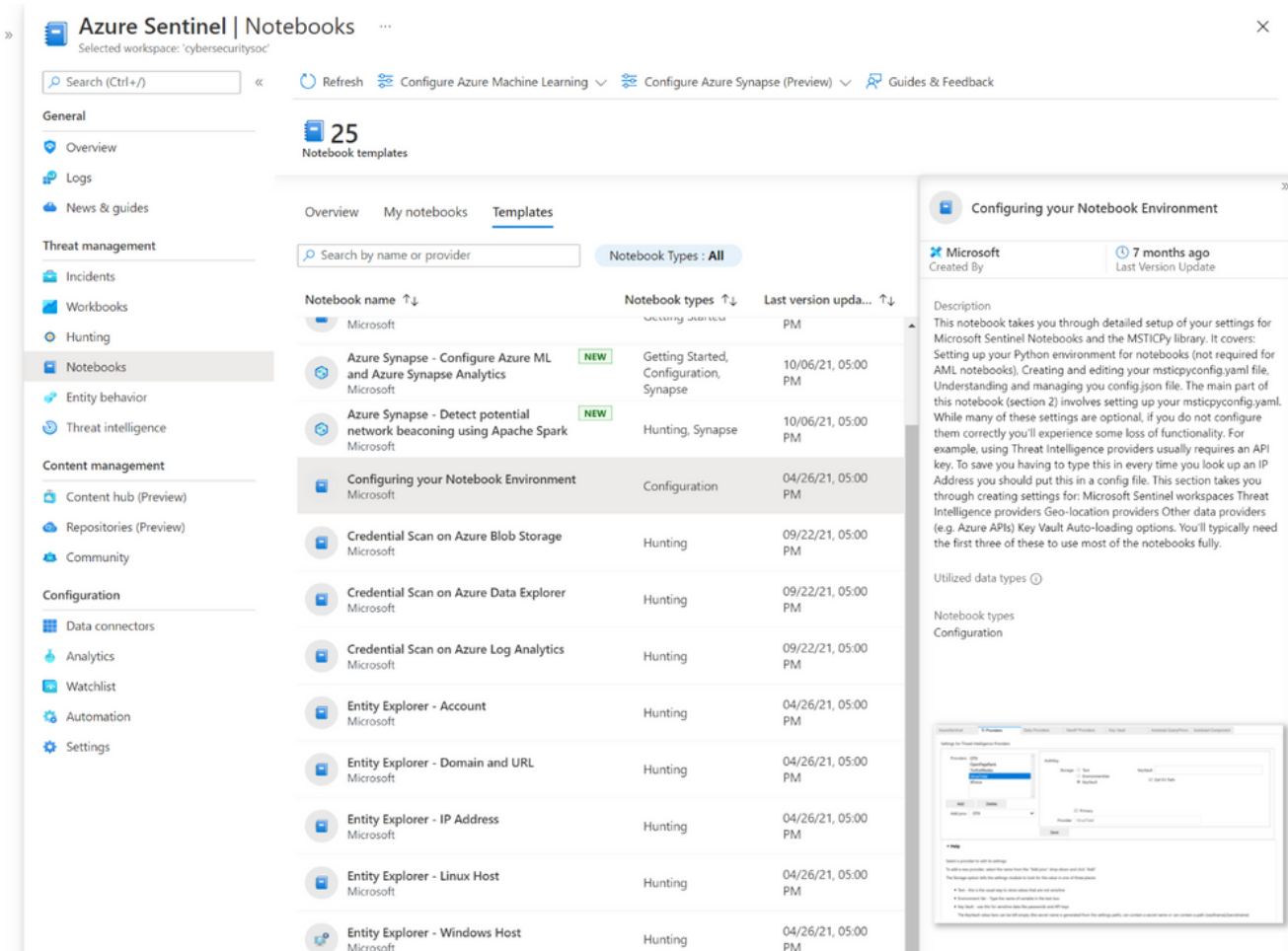
Help

Conf File:

Buttons: Save Settings,  Create backup, Validate Settings

**Note:** If using Azure Machine Learning then you may notice this config widget can take some time to load. We are working to improve this but if you run the notebook in Jupyter, JupyterLab or VSCode you will not have these performance issues.

We have also created a Notebook to help you create to file. Once you have run the 'Getting Started' Notebook it is recommended that you run the 'Configuring your Notebook Environment' Notebook before creating your first Notebook, you can find this in the Microsoft Sentinel portal.



Microsoft Sentinel Notebook feature blade highlighting the Configuring you Notebook Environment Notebook

You can also find more documentation on the config file and creation of it, in the [MSTICPy docs](#)

## Getting Data from Microsoft Sentinel

Querying data from Microsoft Sentinel is handled by MSTICPy's `QueryProvider`. The first step is to initialize a QueryProvider and tell it we want to use the Microsoft Sentinel Query provider.

**Note:** MSTICPy contains several QueryProviders for other data sources as well.

The other thing we want to provide the QueryProvider with is some details of the workspace we want to connect to. We *could* do this manually, but it is much easier to get details from the configuration we set up earlier. We can do this with `WorkspaceConfig`

```
from msticpy.nbtools import nbinit
nbinit.init_Notebook(namespace=globals())
qry_prov=QueryProvider("MicrosoftSentinel")
ws_config = WorkspaceConfig(workspace="MyWorkspace")
```

What WorkspaceConfig is doing is creating the connection string used by the QueryProvider. We can see what that connection string looks like with:

```
ws_config.code_connect_str
```

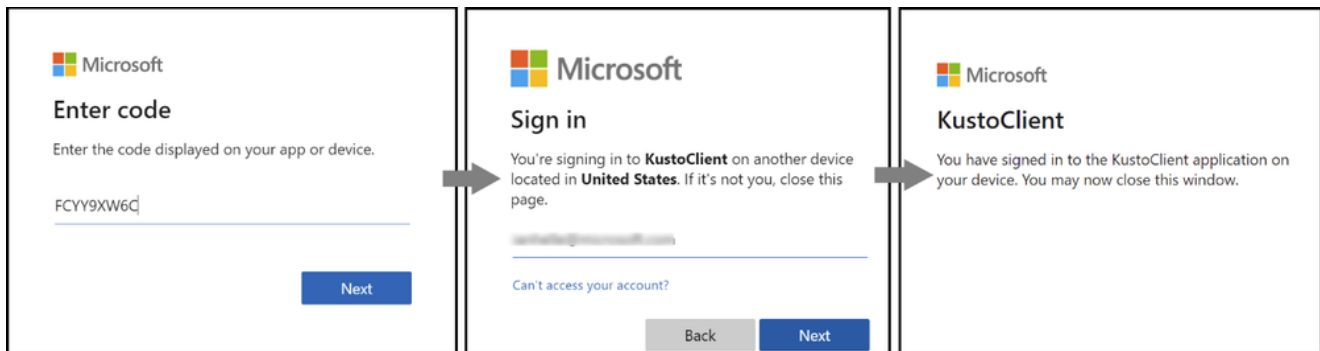
```
1 ws_config.code_connect_str
[65] ✓ <1 sec
... "loganalytics://code().tenant('72f988bf-86f1-41af-91ab-2d7cd011db47').workspace('8ecf8077-cf51-4820-aadd-14040956f35d')"
```

Notebook output showing the connection string generated by code\_connect\_str

Once set up we can tell the `QueryProvider` to `connect` which will kick off the authentication process. There are several ways that we can handle that authentication but when starting off we can use the default options that prompts the user to log in using a Device Code.

```
qry_prov.connect(ws_config)
```

This will then display a code in the Notebook cell output and prompt you to open a browser and end the code shown. You will then login as normal using your Azure AD (Azure Active Directory) credentials.



Screenshots of the Device Code authentication flow

You can then go back to the Notebook and see that the authentication has been completed:

```
M L ↩ ⌵ ...
1 # Get the Azure Sentinel workspace details from msticpyconfig
2 # Loading WorkspaceConfig with no parameters will use the details
3 # of your "Default" workspace (see the Configuring Azure Sentinel settings section earlier)
4 ws_config = WorkspaceConfig()
5
6 # Connect to Azure Sentinel with our QueryProvider and config details
7 qry_prov.connect(ws_config)
✓ 2 min 12 sec
Copy code to clipboard and authenticate here: https://microsoft.com/devicelogin
FCYY9XW6C
popup schema 52b1ab41-869e-4138-9e40-2a4457f09bf0@loganalytics
```

Notebook output showing the completed authentication flow



## Built-in Queries

Now that we are connected to Microsoft Sentinel, we can start to look at running some queries to get some data. MSTICPy comes with several built-in Microsoft Sentinel queries to get some common datasets into the Notebook. These are different to the queries included in the Microsoft Sentinel GitHub and are more focused on collecting common sets of data that users might need to answer analytical questions.

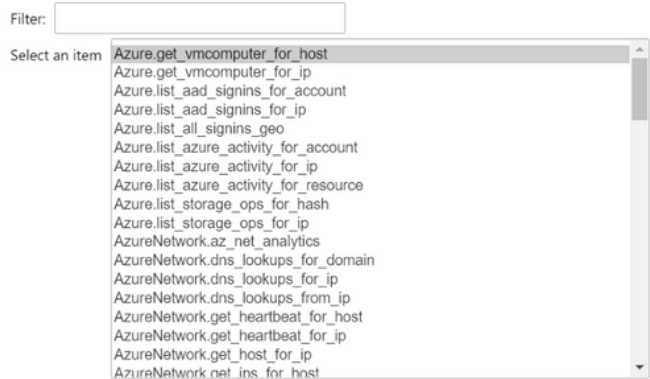
You can see a list of the MSTICPy queries with `.list_queries.`

```
1 qry_prov.list_queries()
[66] ✓ <1 sec
... ['Azure.get_vmcomputer_for_host',
'Azure.get_vmcomputer_for_ip',
'Azure.list_aad_signins_for_account',
'Azure.list_aad_signins_for_ip',
'Azure.list_all_signins_geo',
'Azure.list_azure_activity_for_account',
'Azure.list_azure_activity_for_ip',
'Azure.list_azure_activity_for_resource',
'Azure.list_storage_ops_for_hash',
'Azure.list_storage_ops_for_ip',
'AzureNetwork.az_net_analytics',
'AzureNetwork.dns_lookups_for_domain',
'AzureNetwork.dns_lookups_for_ip',
'AzureNetwork.dns_lookups_from_ip',
'AzureNetwork.get_heartbeat_for_host',
'AzureNetwork.get_heartbeat_for_ip',
'AzureNetwork.get_host_for_ip',
'AzureNetwork.get_ips_for_host',
'AzureNetwork.list_azure_network_flows_by_host',
'AzureNetwork.list_azure_network_flows_by_ip',
'AzureSentinel.get_bookmark_by_id',
'AzureSentinel.get_bookmark_by_name',
'AzureSentinel.list_bookmarks',
'AzureSentinel.list_bookmarks_for_entity',
'AzureSentinel.list_bookmarks_for_tags',
'Heartbeat.get_heartbeat_for_host',
'Heartbeat.get_heartbeat_for_ip',
'Heartbeat.get_info_by_hostname',
'Heartbeat.get_info_by_ipaddress',
...]
```

Notebook output of the `list_queries` command

**Note:** MSTICPy also includes queries for its other Data Providers, and not just Microsoft Sentinel.

You can also use `.browse_queries()` to see the available queries in an interactive browser widget.



---

### Gets latest VMComputer record for Host

#### Parameters

add\_query\_items: str (optional)  
Additional query clauses

end: datetime (optional)  
Query end time

host\_name: str  
The Computer name of the VM

start: datetime (optional)  
Query start time  
(default value is: -5)

table: str (optional)  
Table name  
(default value is: VMComputer)

#### Query

```
{table}
| where TimeGenerated >= datetime({start})
| where TimeGenerated <= datetime({end})
| where Computer has "{host_name}"
| take 1
```

Notebook output of browse\_queries

## Running a query

---

Now that we have found a query that we want to run we simply pass its name to the `QueryProvider` and that in turn returns to results of the query in a Pandas Data Frame. Most queries support additional parameters, but we are showing one here that does not need any parameters.

**Note:** the queries are attached to the QueryProvider as methods (functions) and grouped into categories based on the data source being queried. You can use tab completion or IntelliSense to help you navigate to the query you need.

```
qry_prov.Azure.list_all_signins_geo()
```

```
1 qry_prov.Azure.list_all_signins_geo()
✓ 10 sec
```

	TenantId	SourceSystem	TimeGenerated	ResourceId	OperationName	OperationVersion	Category	ResultType	ResultSignature	ResultDescripti
0	8ecf8077-cf51-4820-aadd-14040956f35d	Azure AD	2021-11-14 03:59:18.995000+00:00	/tenants/4b2462a4-bbee-495a-a0e1-f23ae524cc9c/providers/Microsoft.aadiam	Sign-in activity	1.0	SignInLogs	0	None	
1	8ecf8077-cf51-4820-aadd-14040956f35d	Azure AD	2021-11-14 04:20:18.703000+00:00	/tenants/4b2462a4-bbee-495a-a0e1-f23ae524cc9c/providers/Microsoft.aadiam	Sign-in activity	1.0	SignInLogs	0	None	
2	8ecf8077-cf51-4820-aadd-14040956f35d	Azure AD	2021-11-14 04:23:19.176000+00:00	/tenants/4b2462a4-bbee-495a-a0e1-f23ae524cc9c/providers/Microsoft.aadiam	Sign-in activity	1.0	SignInLogs	0	None	

Output of the list\_all\_signins\_geo query

**Troubleshooting tip:** If a query does not execute at first make sure you have run ``qry_prov.connect()`` to authenticate to Microsoft Sentinel first. Notebook cells do not have to be run in order so you can go back and run any that you missed. However, many notebooks do have cells that rely on previous cells being executed first so be careful about jumping ahead if you have not created the notebook yourself.

**Troubleshooting tip:** If a query is not returning the results you expect, pass 'print' along as a parameter when calling the query to print out the KQL query being executed.

More typically the query function will expect parameters such as the host name or IP address that you are searching for.

```
qry_prov.LinuxSyslog.user_logon(host_name="mylxhost")
```

If you try to run a query without supplying the required parameter, it will return an error message including the help for the query with the parameter definitions.

Most queries also require date/time parameters for the beginning/ending bounds of the query. By default, these are supplied by a time range set in the query provider. Each instance of a query provider has its own time range. You can change the default query range by running the following.

```
qry_prov.query_time
```

This brings up a widget letting you change the defaults for this query provider. You can also supply "start" and "end" parameters to the query function – either as Python datetimes or as time strings:

```
from datetime import datetime
qry_prov.LinuxSyslog.user_logon(
    host_name="mylxhost",
    start="2021-11-19 20:30",
    end=datetime.utcnow()
)
```

## Customizing Your Queries

In addition to the stock query, we can customize certain elements of the query.

For example, if we want to append a line with ``| take 10`` to the query we have selected to limit the number of results returned we can pass that in with the ``add_query_items`` parameter:

```
qry_prov.SecurityAlert.list_alerts(add_query_items="| take 10")
```

```
[76] 1 qry_prov.SecurityAlert.list_alerts(add_query_items="| take 10")
      ✓ <1 sec
```

	TenantId	TimeGenerated	AlertDisplayName	AlertName	Severity	Description	ProviderName	VendorName	VendorOriginalId	SystemAlertId	Res
0	8ecf8077-cf51-4820-aadd-14040956f35d	2021-11-08 21:05:01.755000+00:00	Malicious credential theft tool execution detected	Malicious credential theft tool execution detected	High	A known credential theft tool execution command line was detected.\nEither the process itself or...	MDATP	Microsoft	da637717609891265939_-1692814474	41f0e73a-b20f-c74d-895d-ffc46239c3e	
1	8ecf8077-cf51-4820-aadd-14040956f35d	2021-11-08 21:06:33.977000+00:00	Daily Account Entity Generator	Daily Account Entity Generator	Medium	This pseudo alert rule creates an Account Entity	ASI Scheduled Alerts	Microsoft	aad757ca-c4fa-48b4-a0af-53a796cd084e	877c949d-3fe5-b4cc-5d24-f99892c561a3	
2	8ecf8077-cf51-4820-aadd-14040956f35d	2021-11-08 21:10:09.530000+00:00	Malicious credential theft tool execution detected	Malicious credential theft tool execution detected	High	A known credential theft tool execution command line was detected.\nEither the process itself or...	MDATP	Microsoft	da637716395339341506_2014377908	60899fcc-6069-69fb-5001-f500692be8c3	
3	8ecf8077-cf51-4820-aadd-14040956f35d	2021-11-08 21:44:54.032000+00:00	WAF events	WAF events	High		ASI Scheduled Alerts	Microsoft	9c80d97f-5fa0-420e-94cb-6f455b66c731	d6106fdd-0fef-7fad-39fc-d131dc575f28	

The output of the `list_alerts` query

**Tip:** You can also use [KQLMagic](#) to query Sentinel data using KQL queries within notebooks. [KQLMagic](#) also returns data in a Pandas Data Frame.

## Working With Data

Data returned by the ``QueryProvider`` comes back in a Pandas Data Frame. This provides us with a powerful and flexible way to access our data.

One of the core things we want to do is look at specific rows in our table. Each table has an index that can be used to call a row using ``.loc``, alternatively we can return a row by its position in the table with ``.iloc``

```
alert_df.loc[1]
```

```

1 alert_df.iloc[1]
[79] ✓ <1 sec

...
TenantId 8ecf8077-cf51-4820-aadd-14040956f35d
TimeGenerated 2021-09-29 20:29:48.932000+00:00
DisplayName Maayan Azure Resource PP test
AlertName Maayan Azure Resource PP test
AlertSeverity High
Description
ProviderName ASI Scheduled Alerts
VendorName Microsoft
VendorOriginalId f8b38681-e787-43b3-bbd6-dd0fdf41653b
SystemAlertId 41d2a4c1-7a6b-41b0-8458-3e2550ebef95
ResourceId
SourceComputerId
AlertType 8ecf8077-cf51-4820-aadd-14040956f35d_46e9e770-2cb2-4529-aaaa-ad4a4735480d
ConfidenceLevel
ConfidenceScore NaN
IsIncident False
StartTime 2021-09-15 20:24:44.666000+00:00
EndTime 2021-09-29 20:24:44.666000+00:00
ProcessingEndTime 2021-09-29 20:29:48.932000+00:00
RemediationSteps
ExtendedProperties {"Query Period": "14.00:00:00", "Trigger Operator": "GreaterThan", "Trigger Thre...
Entities [{"id": "3", "ResourceId": "/subscriptions/d1d8779d-38d7-4f06-91db-9cbc8de...
SourceSystem Detection
WorkspaceSubscriptionId d1d8779d-38d7-4f06-91db-9cbc8de0176f
WorkspaceResourceGroup SOC
ExtendedLinks
ProductName
ProductComponentName Azure Sentinel
AlertLink Scheduled Alerts

```

Selecting a row with `iloc`

We can also choose just to return specific columns by providing a list of them to the Data Frame (note the `[:5]` means return the last 5 rows):

```

alert_df.iloc[:5][["AlertName", "AlertSeverity", "Description"]]

```

```

1 alert_df.iloc[:5][["AlertName", "AlertSeverity", "Description"]]
[80] ✓ <1 sec

...

```

	AlertName	AlertSeverity	Description
0	Suspicious administrative activity	Medium	The user "Chris Boehm (chboeh_microsoft.com#ext#@secxpninja.onmicrosoft.com)" performed more th...
1	Maayan Azure Resource PP test	High	
2	Critical IoT Incident	High	
3	Itay - Test	Medium	
4	Malicious credential theft tool execution detected	High	A known credential theft tool execution command line was detected.\nEither the process itself or...

Filtering columns of a DataFrame

We can also do things such as search for rows with specific data:

```

alert_df[alert_df["AlertName"].str.contains("credential theft")]

```

```
[81] 1 alert_df[alert_df["AlertName"].str.contains("credential theft")]
      ✓ <1 sec
```

TenantId	TimeGenerated	DisplayName	AlertName	AlertSeverity	Description	ProviderName	VendorName	VendorOriginalId	SystemAlertId	ResourceId
4	8ecf8077-cf51-4820-aadd-14040956f35d	2021-09-29 21:09:50.512000+00:00	Malicious credential theft tool execution detected	Malicious credential theft tool execution detected	High	A known credential theft tool execution command line was detected.\nEither the process itself or...	MDATP	Microsoft	da637681141320134770_-1707840607	024d715f-a41d-f00f-f5aa-9da0ae979a7a
5	8ecf8077-cf51-4820-aadd-14040956f35d	2021-09-29 21:09:50.456000+00:00	Malicious credential theft tool execution detected	Malicious credential theft tool execution detected	High	A known credential theft tool execution command line was detected.\nEither the process itself or...	MDATP	Microsoft	da637679592297105614_-2008020024	5edd99a8-70dc-f33f-2e3a-252c7c4fa4be

Searching for rows of a DataFrame matching a criteria

**Tip:** Pandas has loads and loads of features to help you find, analyze, transform, and visualize data. As Pandas data structures are key to Microsoft Sentinel Notebooks, we recommended you spend some time getting familiar with some of their features they offer - <https://pandas.pydata.org/>

## Enriching data using external data sources

One of the powerful elements of Notebooks is combining data from Microsoft Sentinel with data from other sources. One of the most common sources of this data in security is Threat Intelligence (TI) data. MSTICPy has support for several Threat Intelligence data sources including:

- VirtusTotal
- GreyNoise
- AlienVault OTX
- IBM XForce
- Microsoft Sentinel TI data
- OPR (for PageRank details)
- ToR ExitNode information.

The first step in using these TI sources is to create a `TILookup` object. This can then be used to perform lookups against the various supported providers.

Lookups can be done against individual items via `.lookup_ioc` or against multiple items with `.lookup_iocs` and you can configure things such as which Threat Intelligence sources are used.

```
ti = TILookup()
ti.lookup_iocs(signin_df, obs_col="IPAddress", providers=["GreyNoise"])
```

```

1 ti = TIlookup()
2 ti.lookup_iocs(signin_df, obs_col="IPAddress", providers=["GreyNoise"])
✓ 7 sec

```

Connecting... connected  
Attempting to sign-in with environment variable credentials...  
Using Open PageRank. See <https://www.domcop.com/openpagerank/what-is-openpagerank>

	loc	locType	Safeloc	QuerySubtype	Provider	Result	Severity	Details	RawResult	Reference	Status
1	91.149.134.43	ipv4	91.149.134.43	None	GreyNoise	False	information	Not found.	<Response [404]>	<a href="https://api.greynoise.io/v3/community/91.149.134.43">https://api.greynoise.io/v3/community/91.149.134.43</a>	404
2	81.211.111.100	ipv4	81.211.111.100	None	GreyNoise	False	information	Not found.	<Response [404]>	<a href="https://api.greynoise.io/v3/community/81.211.111.100">https://api.greynoise.io/v3/community/81.211.111.100</a>	404
3	91.149.134.43	ipv4	91.149.134.43	None	GreyNoise	False	information	Not found.	<Response [404]>	<a href="https://api.greynoise.io/v3/community/91.149.134.43">https://api.greynoise.io/v3/community/91.149.134.43</a>	404
4	20.37.213.236	ipv4	20.37.213.236	None	GreyNoise	False	information	Not found.	<Response [404]>	<a href="https://api.greynoise.io/v3/community/20.37.213.236">https://api.greynoise.io/v3/community/20.37.213.236</a>	404
5	81.211.111.100	ipv4	81.211.111.100	None	GreyNoise	False	information	Not found.	<Response [404]>	<a href="https://api.greynoise.io/v3/community/81.211.111.100">https://api.greynoise.io/v3/community/81.211.111.100</a>	404
6	178.153.94.1	ipv4	178.153.94.1	None	GreyNoise	False	information	Not found.	<Response [404]>	<a href="https://api.greynoise.io/v3/community/178.153.94.1">https://api.greynoise.io/v3/community/178.153.94.1</a>	404

### Lookup\_iocs results

To make viewing results easier there is a widget to allow you to interactively browse results:

`ti.browse_results(ti_df)`

1 ti.browse\_results(vt\_df)

[86] ✓ <1 sec

... No results at TI Severities 'warning' or 'high'  
 Displaying only 'information' severity items.

Filter:

Select an item

20.37.213.236	type: ipv4	(sev: information)	providers: ['VirusTotal']
81.211.111.100	type: ipv4	(sev: information)	providers: ['VirusTotal']
91.149.134.43	type: ipv4	(sev: information)	providers: ['VirusTotal']

---

**20.37.213.236**

**Type: 'ipv4', Provider: VirusTotal, severity: information**

**Details**

VirusTotal	
verbose_msg	Missing IP address
response_code	0
positives	0

**Reference:**

<https://www.virustotal.com/vtapi/v2/ip-address/report>

TI results browser widget

## Azure API Access

MSTICPy also has integration with a range of Azure APIs that can be used to retrieve additional information or perform actions such as get Microsoft Sentinel incidents.

```
from msticpy.data.azure_sentinel import AzureSentinel
azs = AzureSentinel()
azs.connect()
azs.get_incident(incident_id = "7c768f11-31f1-46ca-8a5c-
25df2e6b7021", sub_id = "8df49d90-99eb-4c31-985d-
64b3f33caa93", res_grp= "sent", ws_name="workspace")
```



```

1 from msticpy.data.azure_sentinel import AzureSentinel
2
3 azs = AzureSentinel()
4 azs.connect()
5 azs.get_incident(incident_id = "7c768f11-31f1-46ca-8a5c-25df2e6b7021", sub_id = "8df49d90-99eb-4c31-985d-64b3f33caa93", res_grpa= "sent", ws_name="workspac
✓ 2 sec

```

Attempting to sign-in with environment variable credentials...  
Attempting to sign-in with environment variable credentials...

	id	name	etag	type	properties.title	properties.description	properties.severity	propert
0	/subscriptions/d1d8779d-38d7-4f06-91db-9cbc8de0176f/resourceGroups/soc/providers/Microsoft.Opera...	7a4f5e0e-c202-4298-8cb6-e1278500fbc7	*790056a3-0000-0100-0000-6170ab480000"	Microsoft.SecurityInsights/Incidents	Preview: Crypto-mining activity following Impossible travel to atypical locations	This Fusion incident correlates alerts "Impossible travel to atypical locations" detected by "AA...	High	

## Output of Azure APIs

You can find out more about MSTICPy's support for Azure APIs in the documentation: [https://msticpy.readthedocs.io/en/latest/data\\_acquisition/AzureData.html](https://msticpy.readthedocs.io/en/latest/data_acquisition/AzureData.html) & [https://msticpy.readthedocs.io/en/latest/data\\_acquisition/AzureSentinel.html](https://msticpy.readthedocs.io/en/latest/data_acquisition/AzureSentinel.html)

## Visualizations with MSTICPy

The ability to create complex, interactive visualizations is one of the key benefits of Notebooks, allowing analysts to see data in a unique way and use it to identify patterns of anomalies that may not otherwise be possible to identify.

Creating these visualizations from scratch can be quite a complex task and involve a lot of code if starting from nothing. To make the process easier MSTICPy contains several common visualizations work out the box with common data sources from Microsoft Sentinel, and that can quickly and easily be called with minimal code.

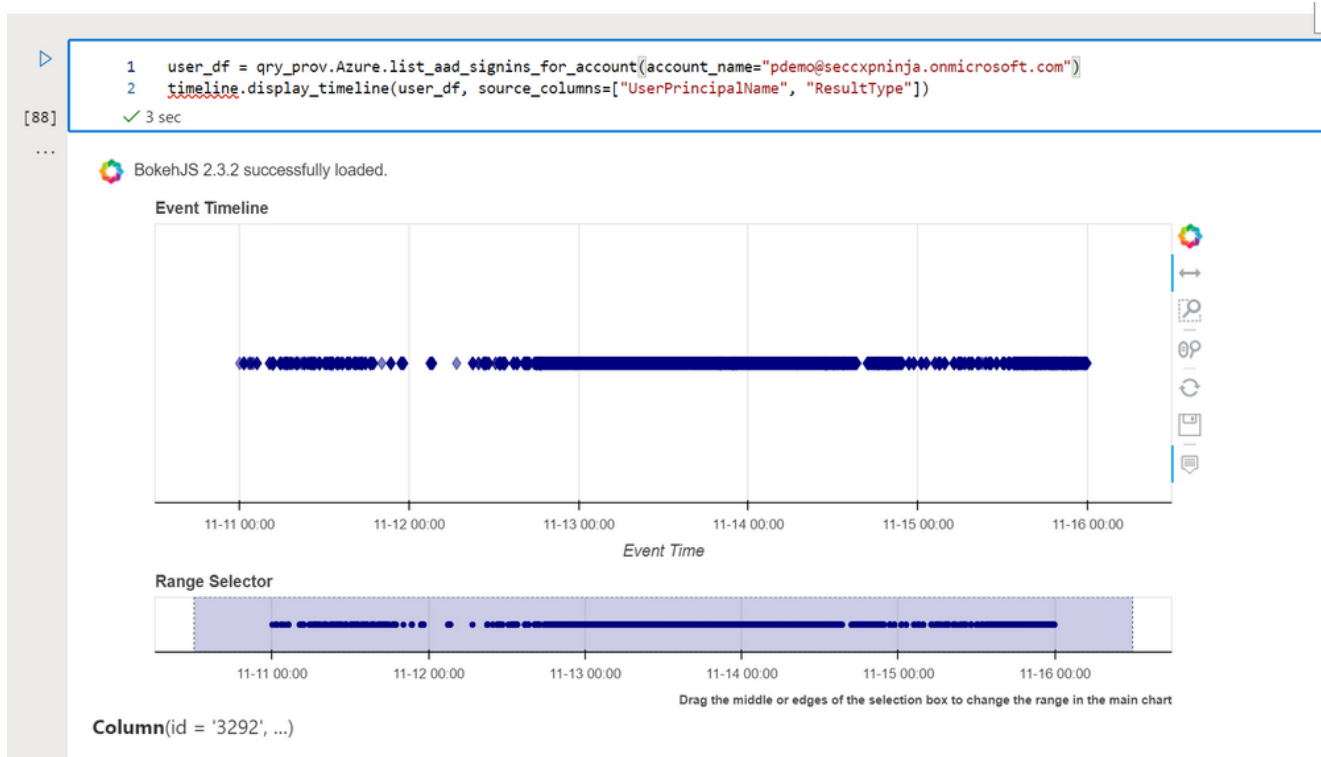
## Timelines

Understanding when events occurred and in what order is a key component of many security investigations. MSTICPy can plot diverse types of timelines with several types of data.

```

user_df = qry_prov.Azure.list_aad_signins_for_account(account_name="pdemo@seccxpinja.c
timeline.display_timeline(user_df, source_columns=["UserPrincipalName", "ResultType"])

```



## Timeline visualization

**Troubleshooting Tip:** If you are defining columns from a DataFrame as a parameter in another function (as we do above with `source_columns`) you can sometimes run into issues if you specify a column that does not exist. If you want to see what columns a DataFrame has you can call `DataFrame.columns` to get a list of all the columns.

We can also plot time lines showing events with a duration rather than a single time stamp with `display_timeline_duration`:

```

timeline_duration.display_timeline_duration(alert_df, group_by="AlertName", time_column

```



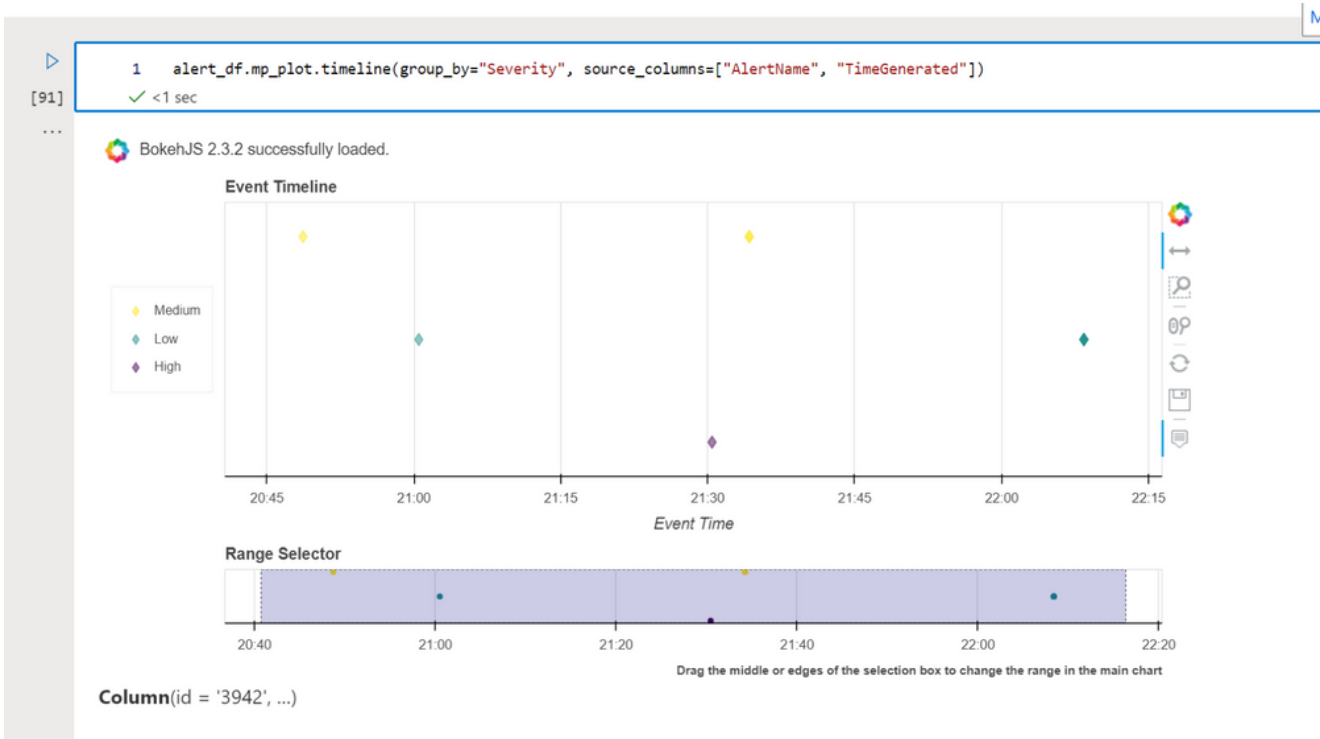
## Timeline duration visualization

**Tip:** You can also call the timeline visualization directly from a DataFrame with 'mp\_plot'

```

alert_df.mp_plot.timeline(group_by="Severity", source_columns=
["AlertName", "TimeGenerated"])

```



## Grouped timeline visualization

## Matrix Plots

The Matrix Plot graph in MSTICPy allows you to plot the interactions between two elements in your data. This can be useful for seeing the relationships between points in a dataset, for example if you wanted to see how often certain IP addresses are communicating with each other in a network you can create a matrix plot with a source IP address on one axis, and a destination IP address on the other axis.

As with the timeline plots, the matrix plot can be created directly from a DataFrame using ``mp_plot``:

```
network_data.mp_plot.matrix(x="SourceIP", y="DestinationIP", title="IP Interaction")
```



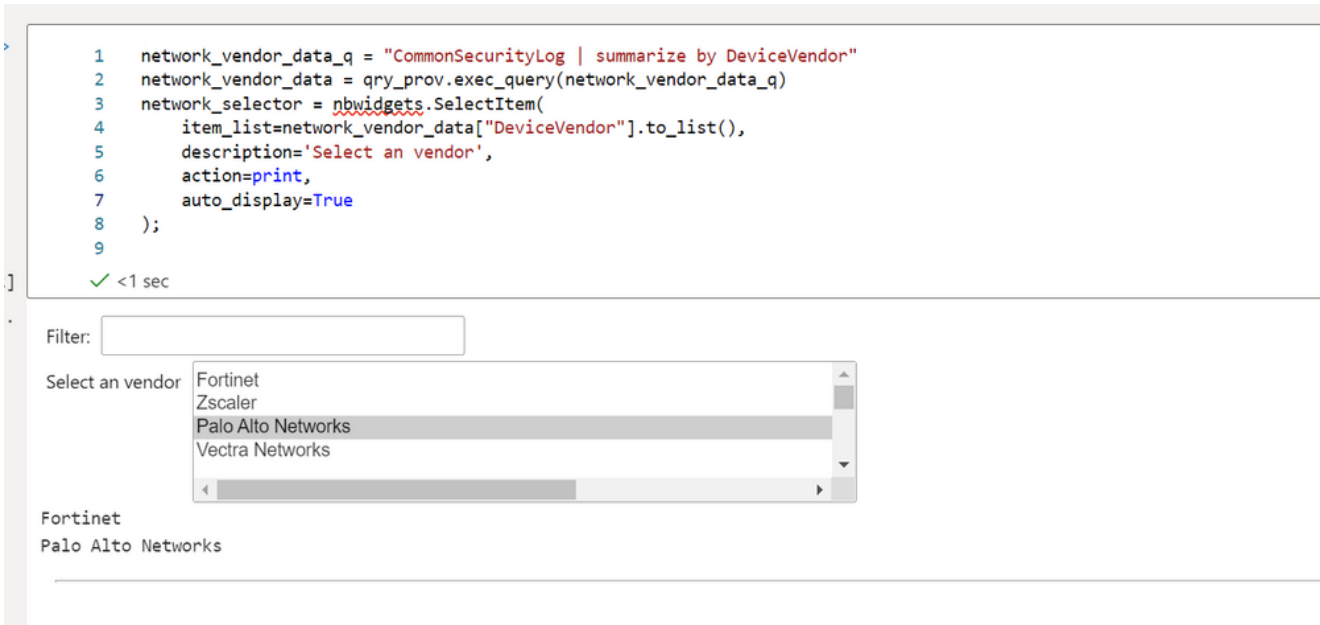
Matrix visualization

## Widgets

We have seen a couple of widgets already in the query and threat intelligence result browsers. These widgets make Notebooks much more accessible by providing a visual way to interact and customize them without having to write any code. MSTICPy includes a

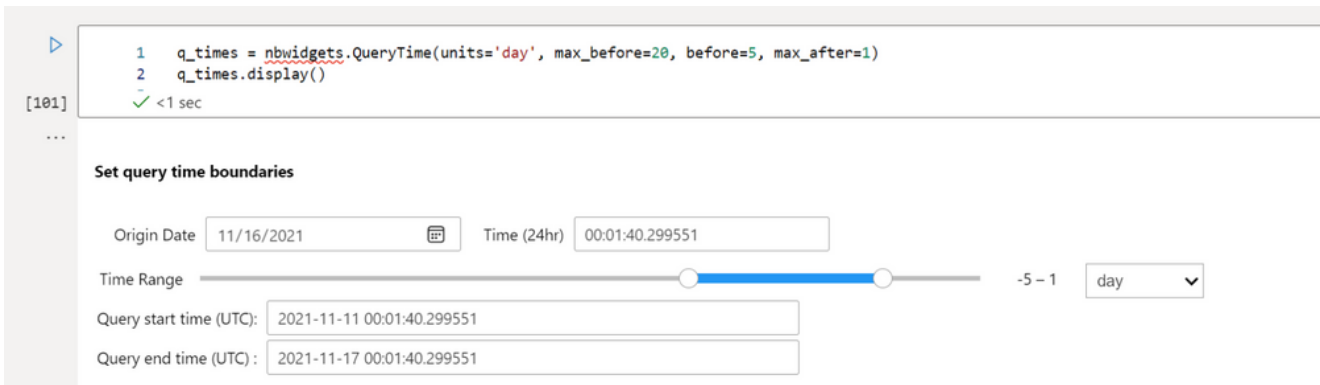
number visual, interactive widgets to allow users to select various parameters to customize the Notebook.

```
network_vendor_data_q = "CommonSecurityLog | summarize by DeviceVendor"  
network_vendor_data = qry_prov.exec_query(network_vendor_data_q)  
network_selector = nbwidgets.SelectItem(  
    item_list=network_vendor_data["DeviceVendor"].to_list(),  
    description='Select a vendor',  
    action=print,  
    auto_display=True  
);
```



Using the SelectItem widget to select a network vendor from data

```
q_times = nbwidgets.QueryTime(units='day', max_before=20, before=5, max_after=1)  
q_times.display()
```



Time range selection widget

```

security_alerts = qry_prov.SecurityAlert.list_alerts(add_query_items="| take 10")
alert_select = nbwidgets.SelectAlert(alerts=security_alerts, action=nbdisplay.display_a

display(Markdown('### Alert selector with action=DisplayAlert'))
display(HTML("<b> Alert selector with action=DisplayAlert </b>"))
alert_select.display()

```

The screenshot shows a Jupyter Notebook cell with the following code:

```

1 security_alerts = qry_prov.SecurityAlert.list_alerts(add_query_items="| take 10")
2 alert_select = nbwidgets.SelectAlert(alerts=security_alerts, action=nbdisplay.display_alert)
3 display(Markdown('### Alert selector with action=DisplayAlert'))
4 display(HTML("<b> Alert selector with action=DisplayAlert </b>"))
5 alert_select.display()

```

The output of the cell is:

**Alert selector with action=DisplayAlert**

Filter alerts by title:

Select alert:

- 2021-11-15 15:59:17.026000+00:00 - Users with Greater Than 1 City - () - [id:375751ee-bda1-a2b6-38c9-a34ceb64976d]
- 2021-11-15 20:58:54.028000+00:00 - Impossible Travel Activity by User - () - [id:63b9dce5-4c9e-0233-ac41-f8d23e58b937]
- 2021-11-15 21:04:42.021000+00:00 - Device is silent - () - [id:3c1c2a0e-5668-f199-28dc-5d531e3fd7c4]
- 2021-11-15 21:04:42.021000+00:00 - Device is silent - () - [id:1f5f855b-8d67-89c5-6e94-12ba1c31e955]
- 2021-11-15 21:04:42.021000+00:00 - Device is silent - () - [id:bffa9c08-dafb-dafb-6bf3-5586ecabe5fe]
- 2021-11-15 21:04:42.021000+00:00 - Device is silent - () - [id:0cfb48e4-eeb2-1531-c6d7-c37b6da065ca]
- 2021-11-15 21:04:42.021000+00:00 - Device is silent - () - [id:fe4c04be-b976-b234-6633-edd1d33e0a43]
- 2021-11-15 21:04:42.021000+00:00 - Device is silent - () - [id:9c3ee153-eb29-7339-5ade-210eb9ca7ecd]
- 2021-11-15 21:04:42.021000+00:00 - Device is silent - () - [id:16ebe92c-005e-4c70-249d-a02634b2234a]
- 2021-11-15 21:04:42.021000+00:00 - Device is silent - () - [id:9b401cce-9033-258d-2101-ec8141b5aad3]

**Alert selector with action=DisplayAlert**

**Selected Alert: 'Users with Greater Than 1 City'**

**Alert\_time:** 2021-11-15 15:59:17.026000+00:00, **Compr\_entity:**, **Alert\_id:** 375751ee-bda1-a2b6-38c9-a34ceb64976d

0	
TenantId	8ecf8077-cf51-4820-aadd-14040956f35d
TimeGenerated	2021-11-15 21:04:53.506000+00:00
AlertDisplayName	Users with Greater Than 1 City
AlertName	Users with Greater Than 1 City
Severity	High
Description	
ProviderName	ASI Scheduled Alerts
VendorName	Microsoft

Alert selector widget

## What to do Next

What you have seen here is just a tiny taster of what Microsoft Sentinel Notebooks can do. However, luckily, we have a lot of additional resources to help you learn what you need and get started with Notebooks.

We recommend that you do the following:

- Sign up for the webinar below where we will cover the topics in this blog in an interactive manner, where you can see the code being executed and learn some extra hints and tips about running Notebooks.

- December 16th 2021 - Become a Jupyter Notebooks Ninja – MSTICPy Fundamentals to Build Your Own Notebooks - [Sign Up Here](#)
- Run the Getting Started Notebook in Microsoft Sentinel
  - This will help you get your config set up
  - This [Documentation](#) will help you in running this notebook
  - There is also an online [tutorials](#)
- Try the interactive MSTICPy Lab – <https://aka.ms/msticpy-demo>
- Go and read the MSTICPy docs – <https://msticpy.readthedocs.io/en/latest/GettingStarted.html>
- Learn more about Pandas - <https://pandas.pydata.org/docs/>
- Check out our other Notebooks for ideas! - <https://github.com/Azure/Azure-Sentinel-Notebooks>